```
import numpy as np
import pandas as pd
from sklearn.preprocessing import Normalizer
from sklearn.cluster import KMeans
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
df = pd.read_csv('/content/Country-data.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     167 non-null    object
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

```
dataframe = df.copy()
dataframe.drop(columns=['country'], inplace=True)
dataframe
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
...	...	...	...	...	...	...	...	...	...
162	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970
163	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500
164	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310
165	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310
166	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460

167 rows × 9 columns

```
values = Normalizer().fit_transform(dataframe.values)
print(values)
```

```
[[5.28625544e-02 5.86059362e-03 4.44232996e-03 ... 3.29365361e-02
  3.41086549e-03 3.24090827e-01]
 [1.54565929e-03 2.60713615e-03 6.09883634e-04 ... 7.10444600e-03
  1.53634809e-04 3.80828101e-01]
 [2.00006203e-03 2.81327406e-03 3.05503980e-04 ... 5.60456942e-03
  2.11728178e-04 3.26750061e-01]
 ...
 [4.97959888e-03 1.53876017e-02 1.46182216e-03 ... 1.56226900e-02
  4.16747546e-04 2.79968864e-01]
 [1.20589885e-02 6.42574875e-03 1.10951262e-03 ... 1.44579347e-02
  1.00027489e-03 2.80591029e-01]
 [2.31349866e-02 1.03007762e-02 1.63977221e-03 ... 1.44767666e-02
  1.50335653e-03 4.06463062e-01]]
```

```
def clustering_algorithm(n_clusters, dataset):
    kmeans = KMeans(n_clusters=n_clusters, n_init=10, max_iter=300)
    labels = kmeans.fit_predict(dataset)
    s = metrics.silhouette_score(dataset, labels, metric='euclidean')
    dbs = metrics.davies_bouldin_score(dataset, labels)
    calinski = metrics.calinski_harabasz_score(dataset, labels)
    return s, dbs, calinski
```

```
for i in range(3, 11):
    s, dbs, calinski = clustering_algorithm(i, values)
    print(i, s, dbs, calinski)
```

```
3 0.5198837827909313 0.6008669317607285 458.31264276466067
4 0.4634990957986046 0.7218455631804814 439.8019905572253
5 0.43859967605023265 0.7710112898249146 417.24674177320094
6 0.4696101045315829 0.7520661493821988 442.4034615165842
7 0.4645904730917045 0.6773860500589699 457.84977996199217
8 0.425997367740665 0.7319353703488833 465.1721966231241
9 0.43663653633042926 0.7353285280488965 468.87261942843736
10 0.43579828501773965 0.6633562896255003 492.7731886302295
```

```
random_data = np.random.rand(167,9)
s_random, dbs_random, calinski_random = clustering_algorithm(3, random_data)
s, dbs, calinski = clustering_algorithm(3, values)

print(s_random, dbs_random, calinski_random)
print(s, dbs, calinski)
```

```
0.09288154412420664 2.518987701787166 17.920394992597448
0.5198837827909313 0.6008669317607285 458.3126427646605
```

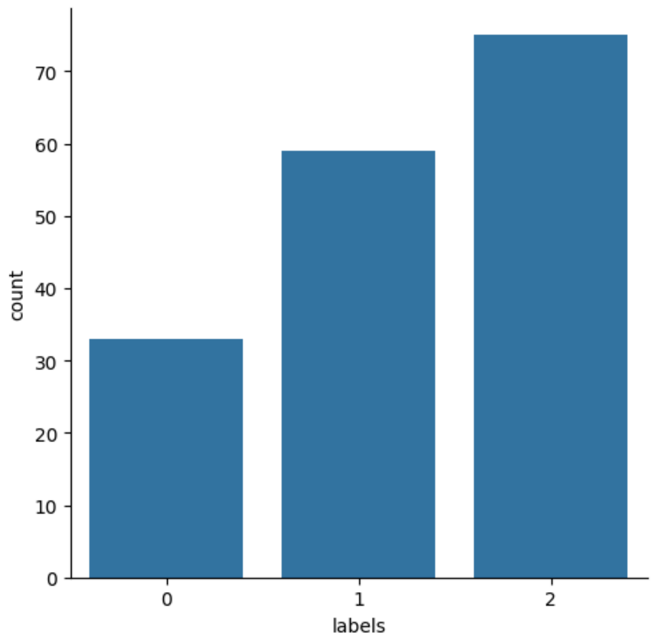
```
set1, set2, set3 = np.array_split(values, 3)
s1, dbs1, calinski1 = clustering_algorithm(3, set1)
s2, dbs2, calinski2 = clustering_algorithm(3, set2)
s3, dbs3, calinski3 = clustering_algorithm(3, set3)
print(s1, dbs1, calinski1)
print(s2, dbs2, calinski2)
print(s3, dbs3, calinski3)
```

```
0.5099002406186719 0.617677169564452 163.35312834956085
0.5355580436538645 0.5880283946904726 188.30786716669212
0.5657004742226224 0.5356671502718732 142.91946826594048
```

```
kmeans = KMeans(n_clusters=3, n_init=10, max_iter=300)
y_pred = kmeans.fit_predict(values)
labels = kmeans.labels_

df['labels'] = labels
```

```
sns.catplot(x='labels', kind='count', data=df)
```

<seaborn.axisgrid.FacetGrid at 0x7e367bb27760>




```
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[1.89973424e-03 1.41159823e-03 5.18643396e-04 2.90914866e-03
  6.78742436e-01 3.45565877e-04 3.63893463e-03 1.78818699e-04
  7.29344202e-01]
 [1.07197437e-02 5.20213097e-03 9.02207865e-04 7.28342214e-03
  8.63230309e-01 1.23833105e-03 9.06304745e-03 6.08413040e-04
```

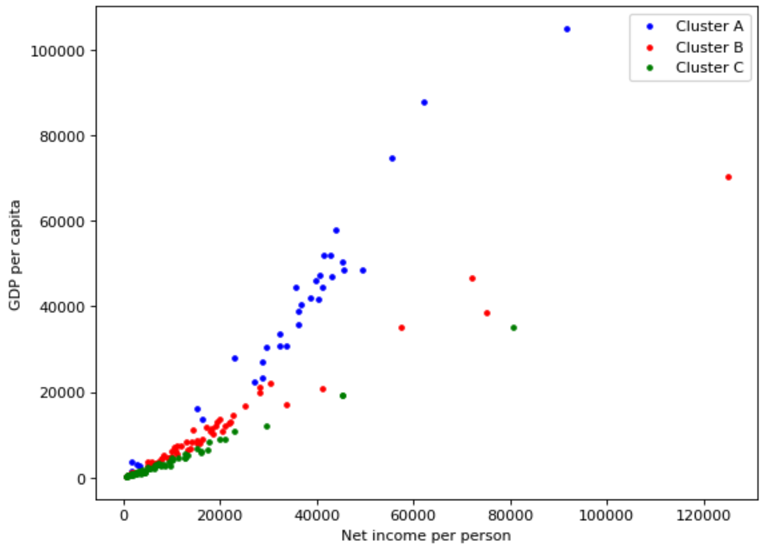
```
 4.99951321e-01]
[2.64610171e-02 8.03462903e-03 2.14032264e-03 1.39988062e-02
 9.31344877e-01 2.92813714e-03 1.95943709e-02 1.48105369e-03
 3.56104512e-01]]
```

```
max = len(centroids[0])
for i in range(max):
    print(dataframe.columns.values[i],"\n{:.4f}".format(centroids[:, i].var()))
```

```
child_mort
0.0001
exports
0.0000
health
0.0000
imports
0.0000
income
0.0114
inflation
0.0000
life_expec
0.0000
total_fer
0.0000
gdpp
0.0236
```

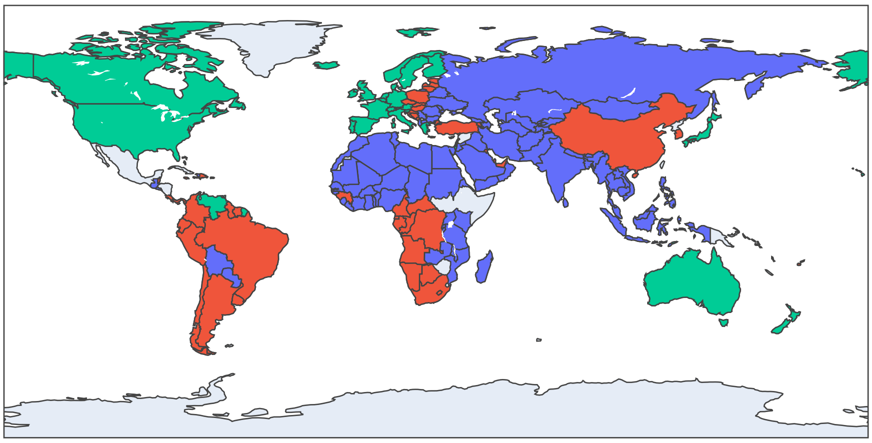
```
df_0 = df[df['labels'] == 0]
df_1 = df[df['labels'] == 1]
df_2 = df[df['labels'] == 2]
```

```
plt.figure(figsize=(8, 6), dpi=80)
plt.scatter(df_0['income'], df_0['gdpp'], c='blue', s=10, label='Cluster A')
plt.scatter(df_1['income'], df_1['gdpp'], c='red', s=10, label='Cluster B')
plt.scatter(df_2['income'], df_2['gdpp'], c='green', s=10, label='Cluster C')

plt.xlabel('Net income per person')
plt.ylabel('GDP per capita')
plt.legend(),
plt.show()
```




```
clusters_name = {0: 'Cluster A', 1: 'Cluster B', 2: 'Cluster C'}
df['labels'] = df['labels'].map(clusters_name)

fig = px.choropleth(df,
                    locationmode='country names',
                    locations='country',
                    color='labels',
                    title='Coutries by labels'
                   )
fig.show()
```

Coutries by labels



```
description = df.groupby("labels")[['child_mort', 'exports', 'health', 'imports', 'income', 'inflation', 'life_expec', 'total_fer', 'gd
n_clients = description.size()
description = description.mean()
description['n_clients'] = n_clients
print(description)
```

```
           child_mort    exports    health    imports        income  \
labels
Cluster A   10.545455  42.875758  9.866667  45.257576  34696.060606
Cluster B   31.310169  49.176271  6.318644  53.616949  18212.322034
Cluster C   55.944000  33.985320  5.864267  42.316879   8582.213333

           inflation  life_expec  total_fer          gdpp  n_clients
labels
Cluster A   3.503455   78.403030   2.103333  38552.121212         33
Cluster B   5.953746   71.157627   2.650678  10734.186441         59
Cluster C  11.102413   66.629333   3.553467   3459.693333         75
```

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster import hierarchy

df = pd.read_csv('/content/Country-data.csv')
print(df.shape)
df = df[['imports','exports', 'health']]
df = df.dropna(axis=0)
clusters = hierarchy.linkage(df, method="ward")

plt.figure(figsize=(8, 6))
dendrogram = hierarchy.dendrogram(clusters)
plt.axhline(100, color='red', linestyle='--');
plt.axhline(100, color='crimson');
```

```
(167, 10)
```